

URZĄD PATENTOWY RZECZYPOSPOLITEJ POLSKIEJ



ZAŚWIADCZENIE

Advanced Digital Broadcast Ltd
Taipei, Tajwan

Advanced Digital Broadcast Polska Sp. z o.o.
Zielona Góra, Polska


złożyli w Urzędzie Patentowym Rzeczypospolitej Polskiej dnia 14 lutego 2003 r. podanie o udzielenie patentu na wynalazek pt. „Odbiornik sygnału danych programowany programem ładującym i sposób aktualizacji oprogramowania w odbiorniku sygnału danych za pomocą programu ładującego. ”

Dołączone do niniejszego zaświadczenia opis wynalazku , zastrzeżenia patentowe i rysunki są wierną kopią dokumentów złożonych przy podaniu w dniu 14 lutego 2003 r

Podanie złożono za numerem P-358733.

Warszawa, dnia 20 marca 2003 r.

z upoważnienia Prezesa


mgr inż. Monika Więckowska
Dyrektor Departamentu

Odbiornik sygnału danych programowany programem ładującym i sposób aktualizacji oprogramowania w odbiorniku sygnału danych za pomocą programu ładującego

- 5 Przedmiotem wynalazku jest odbiornik sygnału danych programowany programem ładującym i sposób aktualizacji oprogramowania w odbiorniku sygnału danych za pomocą programu ładującego, który może być wykorzystywany przykładowo w odbiornikach sygnału radiowego, telewizyjnego lub telefonii komórkowej, a zwłaszcza w dekodernach telewizji cyfrowej.
- 10 Program ładujący o nazwie angielskiej *Loader* obsługuje procedurę aktualizacji oprogramowania. Przetwarza odbierane przez urządzenie pakiety danych i tworzy program, który jest zapisywany w pamięci FLASH urządzenia.
- Stan techniki zawiera projekt nazwany *EUROLOADER*, w którym jest zawarta specyfika programu ładującego dla dekodernach telewizji cyfrowej
- 15 przyjęta przez firmy zrzeszone w organizacji o angielskiej nazwie *European Cable Communication Association*, w skrócie ECCA. Projekt ten opisuje sposób działania programu ładującego, który definiuje procedurę aktualizacji oprogramowania w dekodernach. Według specyfikacji dane dotyczące aktualizacji oprogramowania są często transmitowane aby uniknąć opóźnień w
- 20 aktualizacji. Aplikacje zostały tu podzielone na program *EUROLOADER* oraz program operacyjny (aplikację). Ponadto program *EUROLOADER* jest podzielony na dwie części: kod startera i loadera. Specyfikacja programu *EUROLOADER* wymaga od urządzenia aby było wyposażone w co najmniej cztery rodzaje pamięci, a mianowicie pamięć typu RAM, FLASH, ROM i NV-
- 25 RAM. W pamięci typu RAM są przechowywane informacje tymczasowe, pamięć typu FLASH zawiera kod aplikacji, przy czym możliwa jest aktualizacja danych zapisanych w pamięci typu FLASH. Z kolei w pamięci typu ROM są

przechowywane dane nie podlegające aktualizacji, w tym część kodu programu EUROLOADERA, którym jest kod startera. Pamięć typu NV-RAM jest
30 przeznaczona głównie na informacje o konfiguracji urządzenia.

Aktualny stan techniki znany z komputerów PC zawiera także programy służące do kompresji plików wykonywalnych z rozszerzeniem „.exe”. Ich zadaniem jest kompresja programów w oparciu o znane rozwiązania samorozpakowujących się archiwów, które po dekompresji plików na dysk
35 uruchamiają właściwy program.

Istotą wynalazku jest to, że w odbiorniku sygnału danych programowanym programem ładującym zawierającym blok odbioru sygnału, procesor, interfejsy, pamięć typu RAM, ROM, NV-RAM oraz pamięć nieulotną, procesor posiada wewnętrzny blok inicjujący pracę programu ładującego oraz
40 blok obsługi programu ładującego, który na podstawie kodu zainicjowanego przez blok inicjujący, steruje pracą programu ładującego, który jest zapisany jako skompresowany w pamięci nieulotnej, a po zdekompresowaniu jest zapisany w części obszaru pamięci typu RAM zadeklarowanej jako pamięć typu ROM.

Korzystnie obraz pamięci jest utworzony z części zawierającej sekwencję startową programu ładującego, części zawierającej tablicę skoków programu ładującego oraz części zawierającej obszar statycznych danych programu ładującego oraz kodu programu ładującego i jako skompresowany
45 jest zapisany w pamięci nieulotnej.

Korzystnie tablica skoków programu ładującego zawiera adresy funkcji wspólnych dla programu dekompresującego oraz programu ładującego, które to funkcje zdefiniowane są w programie dekompresującym.

Korzystnie zdekompresowany kod programu ładującego jest umieszczony pod stałym adresem w pamięci RAM.

55 Korzystnie pamięć nieulotna jest pamięcią typu FLASH.

Istotą wynalazku jest również to, że w sposobie aktualizacji oprogramowania w odbiorniku danych za pomocą programu ładującego przechowywanego w pamięci nieulotnej zawierającym blok odbioru sygnału, procesor, interfejsy, pamięć typu RAM, ROM, NV-RAM oraz pamięć nieulotną,

60 oprogramowanie aktualizujące zapisuje się jako skompresowane w pamięci nieulotnej, a podczas uruchomienia procedury startowej kopiuje się je pod stały adres w części obszaru pamięci typu RAM, która przed procesem łączenia została zadeklarowana jako pamięć typu ROM.

Korzystnie procedura startowa programu ładującego wykonywana jest
65 przy włączeniu odbiornika do zasilania.

Korzystnie procedura startowa programu ładującego uruchamiana jest na żądanie użytkownika.

Korzystnie procedura startowa programu ładującego uruchamiana jest przez zewnętrzny sygnał, przesyłany do odbiornika.

70 Korzystnie obraz pamięci tworzy się z części zawierającej sekwencję startową programu ładującego, części zawierającej tablicę skoków programu ładującego oraz części zawierającej obszar statycznych danych programu ładującego oraz kodu programu ładującego a następnie obraz pamięci kompresuje się.

75 Korzystnie tworzy się tablicę zawierającą adresy funkcji wspólnych dla programu dekompresującego oraz programu ładującego, które to funkcje zdefiniowane są w programie dekompresującym.

Korzystnie program ładujący rozpoczyna procedurę aktualizacji aplikacji sprawdzając czy aktualnie nadawany w strumieniu program jest przeznaczony
80 dla odbiornika, z którego został uruchomiony program ładujący, co stanowi warunek aktualizacji aplikacji.

Przedmiot wynalazku jest uwidoczniony w przykładzie wykonania na rysunku, na którym fig. 1A przedstawia dekodery telewizji cyfrowej w uproszczonej wersji, fig. 1B przedstawia telefon komórkowy w uproszczonej
85 wersji, fig. 2 przedstawia organizację pamięci typu FLASH, fig. 3A przedstawia algorytm uruchamiania nieskompresowanego programu ładującego, fig. 3B przedstawia algorytm uruchamiania skompresowanego programu ładującego, fig. 4 przedstawia organizację pamięci typu RAM dla skompresowanego programu ładującego, fig. 5 przedstawia procedurę tworzenia
90 skompresowanego obrazu pamięci, fig. 6 przedstawia procedurę dekompresji

(rozpakowania) programu ładującego, fig. 7 przedstawia tablicę skoków programu ładującego.

Przedstawiony sposób może być zastosowany w różnego rodzaju odbiornikach sygnału, z których dekodery telewizji cyfrowej został podany
 95 jedynie jako przykład. Do odbiorników takich można również zaliczyć telefony komórkowe, palmtopy, odbiorniki radiowe oraz wszelkie inne urządzenia, w których możliwa jest aktualizacja oprogramowania poprzez odbierany przez nie sygnał zewnętrzny.

Fig. 1 przedstawia w uproszczonej wersji odbiornik sygnału
 100 telewizyjnego, którym jest dekodery telewizji cyfrowej 101. Głównym elementem dekodera telewizji cyfrowej 101 jest procesor 120, który zarządza pracą odbiornika. Procesor zawiera wewnętrzny blok inicjujący 121, który inicjuje pracę programu ładującego oraz blok obsługi programu ładującego 122, który na podstawie kodu zainicjowanego przez blok inicjujący 121, steruje pracą
 105 programu ładującego. Do procesora podawany jest sygnał z bloku odbioru sygnału MPEG 110. Dodatkowo procesor 120 ma możliwość dwukierunkowej wymiany danych poprzez interfejsy zewnętrzne 140. Możliwa jest także wymiana danych z pilotem zdalnego sterowania 131 oraz przekazywanie sygnału audio/wideo 130 do odbiornika, na przykład telewizora. Dekodery
 110 telewizji cyfrowej zawiera także kilka rodzajów pamięci, które są dwukierunkowo połączone z procesorem. Są to pamięci typu FLASH 150 i RAM 160. W pamięciach tych przechowywane są programy sterujące pracą dekodera telewizji cyfrowej, włączając w to skompresowany program ładującego, w którym zapisany jest sposób kontroli pamięci nieulotnej,
 115 przykładowo typu FLASH 150 oraz pamięci typu RAM 160 dekodera. Program ładujący jest programem, który odbiera kod dekodera z zewnętrznych źródeł i wymienia go w dekodery telewizji cyfrowej, zapisując go do pamięci typu FLASH 150 dekodera telewizji cyfrowej. Program ładujący nie zajmuje się odtwarzaniem sygnałów audio lub wideo ale jest w stanie komunikować się z
 120 użytkownikiem urządzenia wyświetlając informacje na ekranie w celu pokazania stanu aktualnie wykonywanego uaktualnienia kodu oprogramowania dekodera.

Innym odbiornikiem, dla którego można zastosować rozwiązanie według wynalazku, jest telefon komórkowy 211. Telefon taki został przedstawiony na fig. 1B. Wewnętrzne elementy 221 i 222 procesora 220, pełnią te same funkcje co w przypadku dekodera telewizji z fig. 1A, podobnie jak pamięci 250 i 260. Zmieniają się natomiast elementy odbiornika pełniące funkcje komunikacji ze źródłami danych oraz komunikacji z użytkownikiem. Do pierwszej grupy układów należą blok nadawczo/odbiorczy sygnału GSM 210 oraz blok interfejsów zewnętrznych 240 na przykład IrDA. Druga grupa układów obejmuje blok sterowania wyświetlaczem telefonu 230 oraz blok sterowania klawiaturą 231. Podobne, oczywiście dla specjalisty różnice, można znaleźć w innych urządzeniach w których zastosowany byłby sposób według wynalazku.

Kod, który jest przechowywany w dekodерze telewizji cyfrowej, jest podzielony na dwie części: kod dekodera oraz program ładujący. Kod dekodera, nazywany także kodem wysokiego poziomu lub aplikacją, jest odpowiedzialny za odbiór, dekodowanie oraz wyświetlanie sygnałów audio, wideo oraz innych komponentów takich jak teletext czy napisy w różnych wersjach językowych.

Program ładujący może być uruchamiany za każdym razem, gdy podłącza się dekodер telewizji cyfrowej do źródła zasilania. Po uruchomieniu program ładujący sprawdza możliwe źródła aplikacji. Jeśli dostępna jest możliwość aktualizacji aplikacji przez zewnętrzne interfejsy dekodera 140, program ładujący sprawdza możliwość aktualizacji i w razie możliwości rozpoczyna wymianę aplikacji. Jeśli natomiast dostępna jest możliwość wymiany aplikacji z użyciem sygnałów nadawanych w sygnale telewizji satelitarnej, kablowej lub naziemnej odbieranym przez blok odbioru sygnału 110, program ładujący ustawia się na odpowiedni strumień danych, używając parametrów zapisanych w pamięci NV-RAM. Program ładujący rozpoczyna procedurę aktualizacji aplikacji sprawdzając czy aktualnie nadawany w strumieniu program jest przeznaczony dla dekodera telewizji cyfrowej, z którego został uruchomiony program ładujący. Jeśli tak, procedura aktualizacji jest akceptowana.

Program ładujący może być uruchamiany również na żądanie
 155 użytkownika lub aktywowany określonym sygnałem ze strony nadawcy.
 Nadawca może przesłać sygnał informujący o dostępnej nowej wersji
 oprogramowania, a użytkownik po odebraniu tej informacji może wydać
 żądanie uruchomienia programu ładującego lub też program ładujący może być
 uruchomiony automatycznie. W sygnale tym mogą być przesłane dodatkowe
 160 informacje określające źródło i typ przesyłanych danych. Zarówno kod
 aplikacji jak i kod programu ładującego są przechowywane w pamięci typu
 FLASH 150 dostępnej w dekodерze telewizji cyfrowej. Fig. 2 przedstawia
 organizację pamięci typu FLASH dekodera przedstawionego na fig. 1. Pamięć
 typu FLASH jest podzielona na cztery części. Pierwsza część 201 zawiera
 165 oprogramowanie dekodera, druga część 202 to wolny obszar pamięci, jeśli taki
 jest dostępny. Następna część 203 zawiera opcjonalne dane aplikacji, ostatnia
 część to kod programu ładującego 204. Przeważnie obszar pamięci dostępny
 dla kodu programu ładującego to 128 kilobajtów. Na początku startu programu
 ładującego, cały jego kod jest kopiowany do pamięci typu RAM 160 dostępnej
 170 w dekodерze telewizji cyfrowej.

Fig. 3A przedstawia algorytm uruchamiania programu ładującego. W
 kroku 301 uruchamiany jest program startujący, który rozpoczyna swoją pracę
 bezpośrednio po włączeniu zasilania dekodera telewizji cyfrowej lub po
 ustawieniu wartości początkowych (restarcie) systemu. Procedura opcjonalnie
 175 inicjuje pamięć typu FLASH zależnie od konfiguracji oraz sprawdza aktualny
 stan dekodera telewizji cyfrowej w celu ustalenia przykładowo czy uruchomić
 program ładujący czy program aplikacji. W kroku 302 wykonywana jest część
 inicjująca, która zależy od sprzętowej konfiguracji dekodera, przykładowo kod
 programu jest kopiowany z pamięci typu FLASH do pamięci typu RAM. W
 180 kroku 303 jest wywoływana funkcja main(), która uruchamia program ładujący,
 a w kroku 304 program ładujący jest realizowany.

Fig. 3B przedstawia algorytm uruchamiania skompresowanego
 programu ładującego według wynalazku, mającego zastosowanie zwłaszcza w
 rozbudowanych konfiguracjach. W niektórych konfiguracjach kod programu
 185 ładującego jest na tyle rozbudowany, że nie można go umieścić w dostępnych

128 kilobajtach w pamięci typu FLASH 150. Aby zmniejszyć objętość programu ładującego stosuje się kompresję jego kodu. Wprowadzenie kompresji kodu programu ładującego wpływa na zwiększenie puli dostępnej pamięci dla przykładowo rozbudowanych sterowników do poszczególnych części dekodera telewizji cyfrowej przykładowo modulatora czy tunera. W tym przypadku kod programu ładującego składa się z: rzeczywistej aplikacji programu ładującego oraz modułu dekompresującego (przywracającego pierwotną strukturę kodu programu ładującego), który rozpakowuje właściwy kod programu ładującego do pamięci typu RAM 160 dostępnej w dekodrze telewizji cyfrowej. Procedura uruchomienia tak skompresowanego programu jest podzielona na dwie części. Pierwsza część, składająca się z kroków 305, 306, 307, 308, 309, 310 jest obsługiwana przez program dekompresujący, a druga część, składająca się z kroku 311 jest obsługiwana przez program ładujący. W kroku 305 uruchamiany jest program startowy po włączeniu zasilania dekodera telewizji cyfrowej lub po ustawieniu wartości początkowych (restart) systemu. W kroku 306 następuje kopiowanie kodu z pamięci typu FLASH do pamięci typu RAM. W kroku 307 następuje dekompresja (rozpakowanie, przywrócenie pierwotnej postaci kodu) programu do pamięci typu RAM dekodera telewizji cyfrowej. W kroku 308 następuje weryfikacja zdekompresowanego programu, czyli kodu w pierwotnej postaci. Następnie w kroku 309 inicjuje się tablicę skoków, która została przedstawiona na fig. 7. W kroku 310 jest wykonywana funkcja `main()`, która uruchamia program ładujący. W kroku 311 następuje uruchomienie programu ładującego.

Nieskompresowane programy ładujące mogą zostać umieszczone w dowolnym obszarze dostępnej pamięci. W przypadku skompresowanego programu ładującego korzysta się ze stałych adresów pamięci, aby poprawne były odwołania do miejsc zawierających zmienne, stałe lub wskaźniki do funkcji. Dlatego, dodatkowo zakłada się, że zdekompresowany kod programu ładującego jest umieszczony pod stałym adresem w pamięci typu RAM. Aby uchronić się przed dodaniem przez program łączący, po angielsku *linker*, procedur, które standardowo są dołączane, gdy kod jest umieszczony w pamięci typu RAM, niezbędne jest podanie programowi łączącemu, że ten

obszar, przeznaczony na program ładujący, to pamięć typu ROM. Podczas
 łączenia procedur przez program łączący następuje zebranie wszystkich
 220 modułów danego programu, uzupełnienie informacji o odwołaniach pomiędzy
 poszczególnymi modułami, przypisanie ostatecznych adresów dla funkcji i dla
 danych, dla których nie zostało to zrobione na etapie kompilacji programu oraz
 wygenerowanie programu wynikowego.

Obsługa skompresowanego programu ładującego wymaga specjalnego
 225 podejścia do operacji umieszczania programów i ich danych w pamięci typu
 FLASH. Przyjmuje się, że adresy pamięci typu RAM zaczynają się od adresu
 0xC0000000 i rozmiar pamięci wynosi osiem megabajtów a pamięci typu
 FLASH cztery megabajty. Zwiększenie ilości dostępnej pamięci typu RAM
 powoduje wzrost ilości dostępnego miejsca dla obszaru ogólnego
 230 przeznaczenia. Adresy pozostałych obszarów są stałe.

Fig. 4 przedstawia szczegółową organizację (podział) pamięci typu
 RAM. Blok 401 reprezentuje obszar zarezerwowany przez program ładujący,
 czyli obszar pamięci zarezerwowany dla sterowników OSGL (skrót od
 angielskich słów *On Screen Graphic for Loader*), odpowiedzialnych za
 235 wyświetlanie elementów na ekranie odbiornika) oraz DMUX (skrót od
 angielskich słów *DeMULTiplexer submodule*), kontrolujący filtrowanie danych
 przychodzących do dekodera telewizji cyfrowej, które są przeznaczone dla
 programu ładującego.

Adres:	0xC0000000
Rozmiar:	256 kB

240

Blok 402 reprezentuje sekwencję startową programu ładującego, to jest
 obszar pamięci, który zawiera blok kodu startowego. Adres pierwszego bajtu z
 obszaru 402 (0xC0040000) jest punktem startowym dla programu ładującego.

245

Adres:	0xC0040000
Rozmiar:	64 kB

Blok 403 reprezentuje obszar pamięci zarezerwowany przez tablicę skoków programu ładującego do funkcji wspólnych dla programu ładującego i modułu dekompresującego (przywracającego pierwotną postać kodu).

Adres:	0xC0040040
Rozmiar:	960 B

Blok 404 reprezentuje obszar pamięci zajęty przez kod programu ładującego po dekompresji oraz przez statyczne dane programu ładującego, przykładowo stałe lub statyczne zmienne.

Adres:	0xC0040400
Rozmiar:	255 kB

Blok 405 reprezentuje zakres pamięci zajęty przez kod programu dekompresującego oraz przez statyczne dane programu dekompresującego, przykładowo stałe lub statyczne zmienne.

Adres:	0xC0080000
Rozmiar:	20 kB

265

Blok 406 reprezentuje obszar ogólnego przeznaczenia zarezerwowany w pamięci. Obszar 406 reprezentuje zakres adresów przeznaczony zarówno na dane programu dekompresującego, w tym skompresowany kod programu ładującego oraz kod modułu programu ładującego.

Wykorzystanie skompresowanego programu ładującego w dekodernach telewizji cyfrowej wymaga wykonania procedury tworzenia obrazu pamięci

RAM. Aby wykonać tę procedurę po kompilacji i łączeniu tworzy się dwa pliki zawierające obraz pamięci. W niektórych systemach są one oznaczone rozszerzeniem „.hex”. Pierwszy zawiera sekwencję startową programu

275 ładującego umieszczoną w bloku 402, drugi to segment pamięci zawierający statyczne dane i kod programu ładującego umieszczone w bloku 404. Bloki 402 i 404 są używane do stworzenia obrazu pamięci typu RAM.

Fig. 5 przedstawia procedurę tworzącą skompresowany obraz pamięci. Po starcie, w kroku 501 są odczytywane pliki z obrazem pamięci. W kroku 502

280 są sprawdzane wejściowe pliki z obrazem pamięci, a mianowicie czy dane statyczne i kod programu ładującego są umieszczone we właściwym przedziale adresów. Jeżeli pliki nie są poprawne, w kroku 506 jest wyświetlana informacja o błędzie i następuje zakończenie procedury. W przypadku, gdy pliki są poprawne, w kroku 503 tworzony jest obraz pamięci typu RAM. Obraz taki

285 stanowi tablicę bajtów złożoną w kolejności z obszarów 402, 403, 404 pamięci. Na etapie tworzenia obrazu pamięci typu RAM, obszar 403 jest wypełniony zerami. W kroku 504 obraz pamięci jest kompresowany, a w kroku 505 jest tworzony plik nagłówkowy programu ładującego zawierający skompresowany obraz pamięci typu RAM oraz informacje potrzebne do poprawnej dekompresji

290 oraz umieszczenia kodu programu ładującego w pamięci typu RAM.

Plik nagłówkowy programu ładującego zawiera następujące opcje dla modułu dekompresującego:

- loader_area_start_addr – adres do miejsca w pamięci, gdzie rozpoczyna się sekwencja startowa programu ładującego – blok 401;
- 295 — loader_area_size – zmienna określająca wielkość bloku dostępnego dla programu ładującego;
- loader_code_size – zmienna określająca rozmiar kodu po wykonaniu operacji kompresji;
- compressed_loader_crc – suma kontrolna skompresowanego kodu programu ładującego;
- 300 — loader_compression – wybrany typ kompresji;
- original_code_addr – adres kodu programu ładującego po dekompresji – blok 404

- original_code_size – rozmiar kodu programu ładującego przed kompresją;
- original_code_crc – suma kontrolna kodu programu ładującego przed kompresją.

Fig. 6 przedstawia procedurę działania programu dekompresującego program ładujący. Plik nagłówkowy programu ładującego jest wykorzystywany przez moduł programu dekompresującego, którego jest częścią. W kroku 601 następuje dekompresja kodu programu ładującego do pamięci typu RAM pod adres 0x0040400. W kroku 602 sprawdza się czy wystąpił błąd podczas dekompresji kodu. W przypadku wystąpienia błędu, następuje przejście do kroku 607. Jeżeli nie ma błędu, to w kroku 603 jest obliczana suma kontrolna CRC zdekompresowanego programu ładującego i następuje przejście do kroku 604, w którym sprawdzane jest, czy suma kontrolna CRC jest poprawna. Gdy suma kontrolna nie jest poprawna, ma miejsce przejście do kroku 607. W przypadku, gdy suma jest poprawna, następuje najpierw przejście do kroku 605, gdzie program dekompresujący inicjalizuje tablicę skoków programu ładującego, która jest przedstawiona na fig. 7, a następnie przejście do kroku 606. Zainicjowanie tablicy skoków jest częścią procesu dekompresującego i wykonywane jest poprzez wypełnianie jej odpowiednimi adresami. W kroku 607 procedura sprawdza, czy kod aplikacji jest poprawny a następnie w kroku 608 sprawdza czy wystąpił błąd. Jeżeli nie wystąpił błąd, to w kroku 609 dla dekodera telewizji cyfrowej ustawiane są parametry początkowe i procedura kończy działanie uruchamiając aplikację. Jeżeli wystąpił błąd, następuje przejście do kroku 610, gdzie procedura ustawia parametry początkowe dla dekodera. Sumy kontrolne liczone są dla obrazów pamięci przed i po kompresji, tak aby program dekompresujący mógł sprawdzić poprawność kodu programu ładującego przed skompresowaniem i po dekompresji.

Fig. 7 przedstawia tablicę skoków programu ładującego. Tablica jest zbiorem adresów, z których każdy zajmuje obszar czterech bajtów. Tablicę rozpoczyna adres bazowy 701 globalnych zmiennych statycznych. Następnymi adresami są adres 702 funkcji inicjującej tablicę wykorzystywaną do liczenia sum kontrolnych CRC, adres 703 funkcji liczącej sumę kontrolną CRC 16,

adres 704 funkcji liczącej sumę kontrolną CRC 32, adres 705 funkcji liczącej sumę kontrolną CRC MPEG32, adres 706 funkcji A do dekompresji programu ładującego oraz adres 707 funkcji B do dekompresji danych aktualizacyjnych. Istnieją dwie funkcje dekompresujące, gdyż program ładujący może być
340 skompresowany inną metodą kompresji, niż dane aktualizacyjne, które są odbierane.

Zastosowanie skompresowanego programu ładującego daje możliwość korzystania z większej ilości sterowników oraz zmniejsza wymaganą ilość pamięci nieulotnej, na przykład typu FLASH, które w przeliczeniu na jednostkę
345 pamięci, są droższe od pamięci typu RAM. W odróżnieniu od rozwiązań znanych z komputerów PC zastosowanie kompresji w urządzeniach, które nie posiadają dysku twardego jest znacznie trudniejsze ze względu na wymóg stworzenia odpowiedniej struktury pamięci typu RAM i wykorzystaniu tej struktury. Ponadto w rozwiązaniu korzysta się z obrazów pamięci a nie z plików
350 wykonywalnych tak jak ma to miejsce w aktualnym stanie techniki.

PEŁNOMOCNIK
HUDY
Dr inż. LUDWIK HUDY
Rzecznik Patentowy
Nr rej. 3098

Zastrzeżenia patentowe

1. Odbiornik sygnału danych programowany programem ładującym zawierający blok odbioru sygnału, procesor, interfejsy, pamięć typu RAM, ROM, NV-RAM oraz pamięć nieulotną znamienny tym, że procesor zawiera wewnętrzny blok (121, 221) inicjujący pracę programu ładującego oraz blok obsługi programu ładującego (122, 222), który na podstawie kodu zainicjowanego przez blok inicjujący (121, 221), steruje pracą programu ładującego, który jest zapisany jako skompresowany w pamięci nieulotnej (150, 250), a po zdekompresowaniu jest zapisany w części obszaru pamięci typu RAM (160, 260) zadeklarowanej jako pamięć typu ROM.
2. Odbiornik sygnału danych według zastrz. 1 znamienny tym, że obraz pamięci jest utworzony z części zawierającej sekwencję startową programu ładującego, części zawierającej tablicę skoków programu ładującego oraz części zawierającej obszar statycznych danych programu ładującego oraz kodu programu ładującego i jako skompresowany jest zapisany w pamięci nieulotnej (150, 250).
3. Odbiornik sygnału danych według zastrz. 1 znamienny tym, że tablica skoków programu ładującego zawiera adresy funkcji wspólnych dla programu dekompresującego oraz programu ładującego, które to funkcje zdefiniowane są w programie dekompresującym.
4. Odbiornik sygnału danych według zastrz. 1 znamienny tym, że zdekompresowany kod programu ładującego jest umieszczony pod stałym adresem w pamięci RAM.

5. Odbiornik sygnału danych według zastrz. 1 znamieny tym, że pamięć nieulotna (150, 250) jest pamięcią typu FLASH.
- 30
6. Sposób aktualizacji oprogramowania w odbiorniku danych za pomocą programu ładującego przechowywanego w pamięci nieulotnej zawierającym blok odbioru sygnału, procesor, interfejsy, pamięć typu RAM, ROM, NV-RAM oraz pamięć nieulotną znamieny tym, że oprogramowanie aktualizujące zapisuje się jako skompresowane w pamięci nieulotnej (150, 250), a podczas uruchomienia procedury startowej kopiuje się je pod stały adres w części obszaru pamięci typu RAM (160, 260), która przed procesem łączenia została zadeklarowana jako pamięć typu ROM.
- 35
7. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz. 6 znamieny tym, że procedura startowa programu ładującego wykonywana jest przy włączeniu odbiornika do zasilania.
- 40
8. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz. 6 znamieny tym, że procedura startowa programu ładującego uruchamiana jest na żądanie użytkownika.
- 45
9. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz. 6 znamieny tym, że procedura startowa programu ładującego uruchamiana jest przez zewnętrzny sygnał, przesyłany do odbiornika.
- 50
10. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz. 6 znamieny tym, że obraz pamięci tworzy się z części zawierającej sekwencję startową programu ładującego, części zawierającej tablicę skoków programu ładującego oraz części zawierającej obszar statycznych danych programu ładującego oraz kodu programu ładującego a następnie obraz pamięci kompresuje się.
- 55

11. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz.
60 6 znamienny tym, że tworzy się tablicę zawierającą adresy funkcji wspólnych
dla programu dekompresującego oraz programu ładującego, które to funkcje
zdefiniowane są w programie dekompresującym.

12. Sposób aktualizacji oprogramowania w odbiorniku danych według zastrz.
65 6 znamienny tym, że program ładujący rozpoczyna procedurę aktualizacji
aplikacji sprawdzając czy aktualnie nadawany w strumieniu program jest
przeznaczony dla odbiornika, z którego został uruchomiony program ładujący,
co stanowi warunek aktualizacji aplikacji.

PEŁNOMOCNIK
Hudy
Dr inż. LUDWIK HUDY
Rzecznik Patentowy
Nr rej. 3098

1 / 7

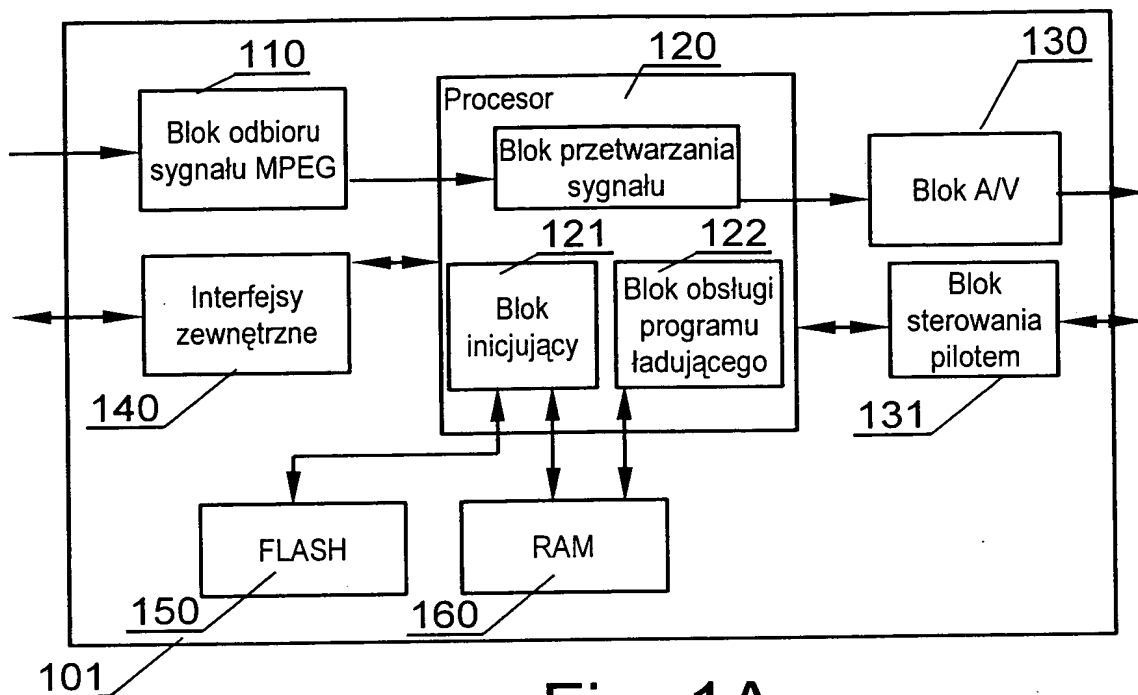


Fig. 1A

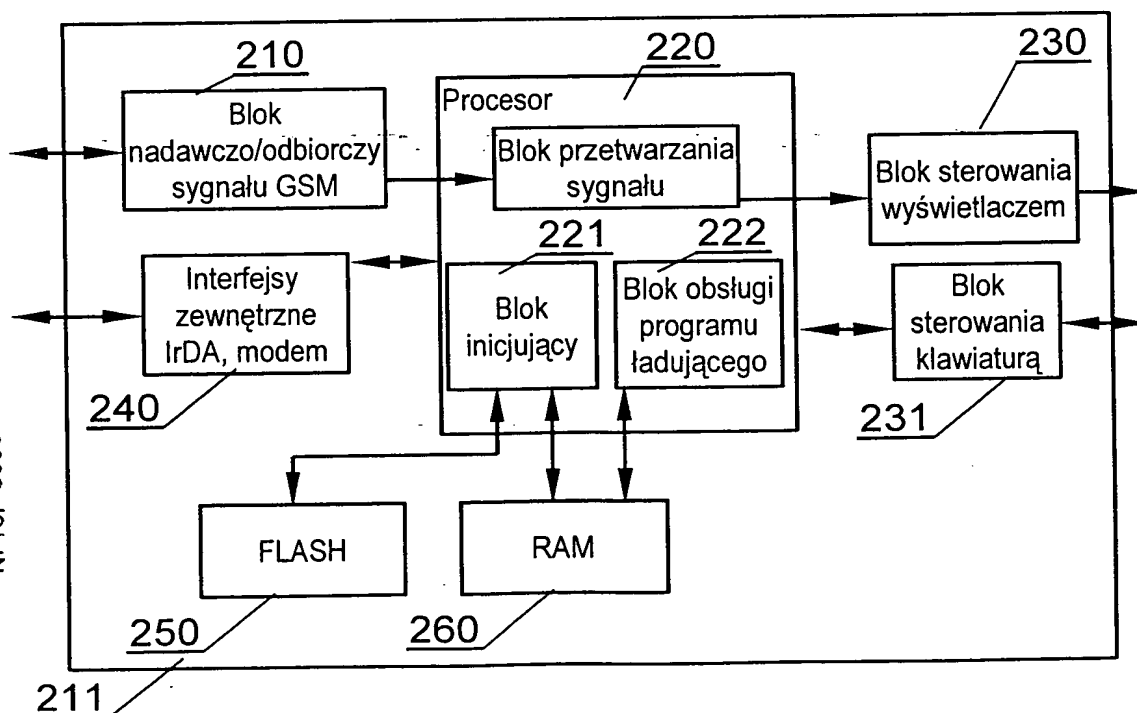


Fig. 1B

6

2 / 7

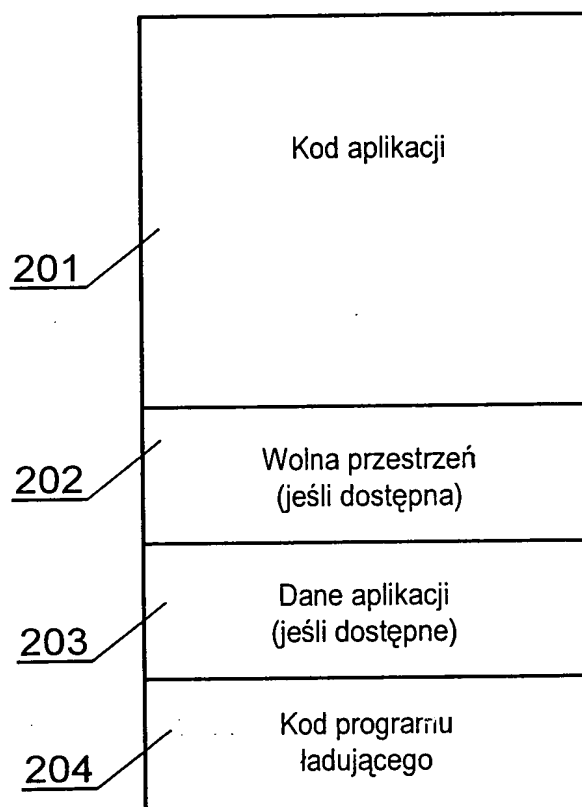


Fig. 2

PEŁNOMOCNIK

HUDY
Dr inż. LUDWIK HUDY
Rzecznik Patentowy
Nr rej. 3098

3 / 7

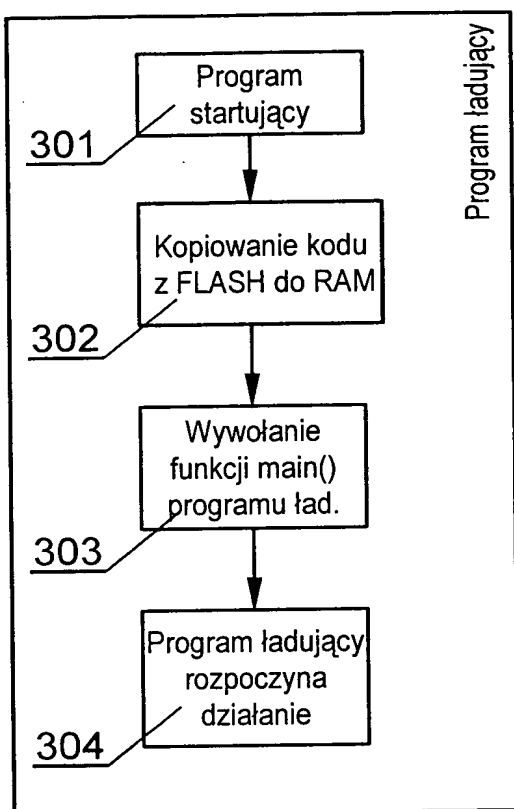


Fig. 3A

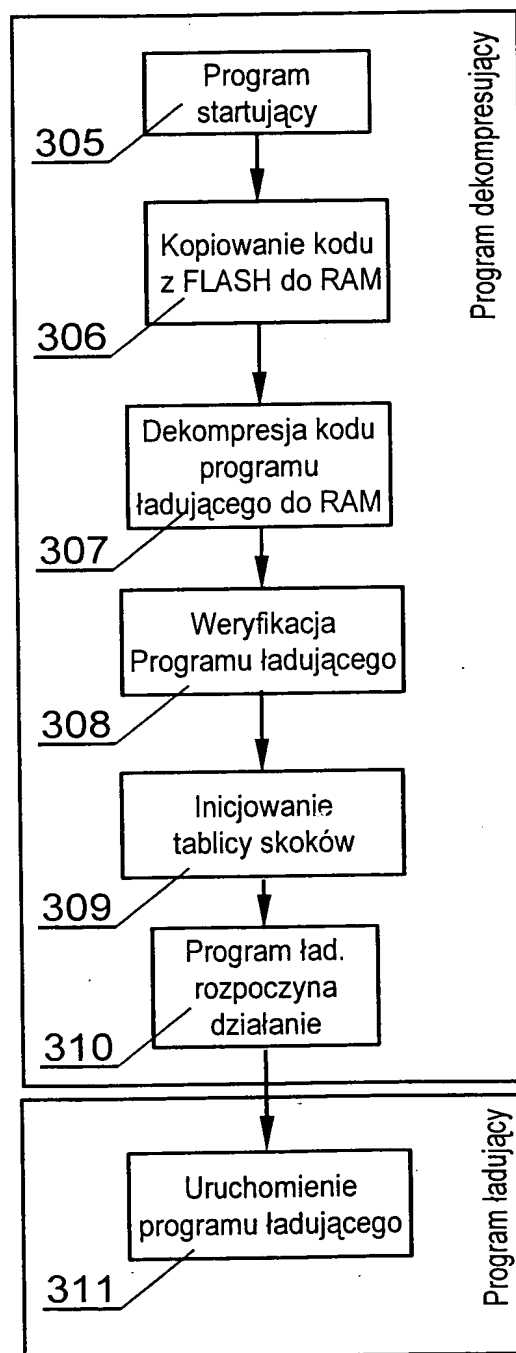


Fig. 3B

4 / 7

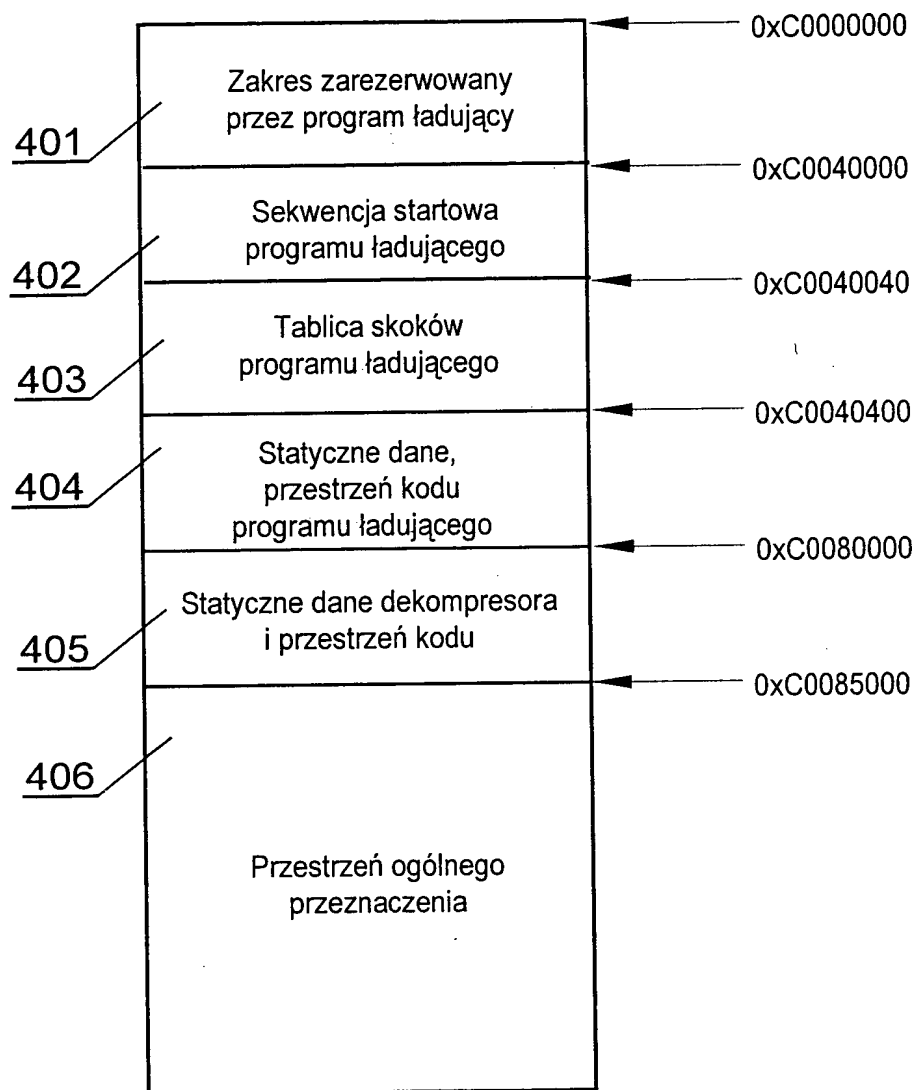


Fig. 4

5 / 7

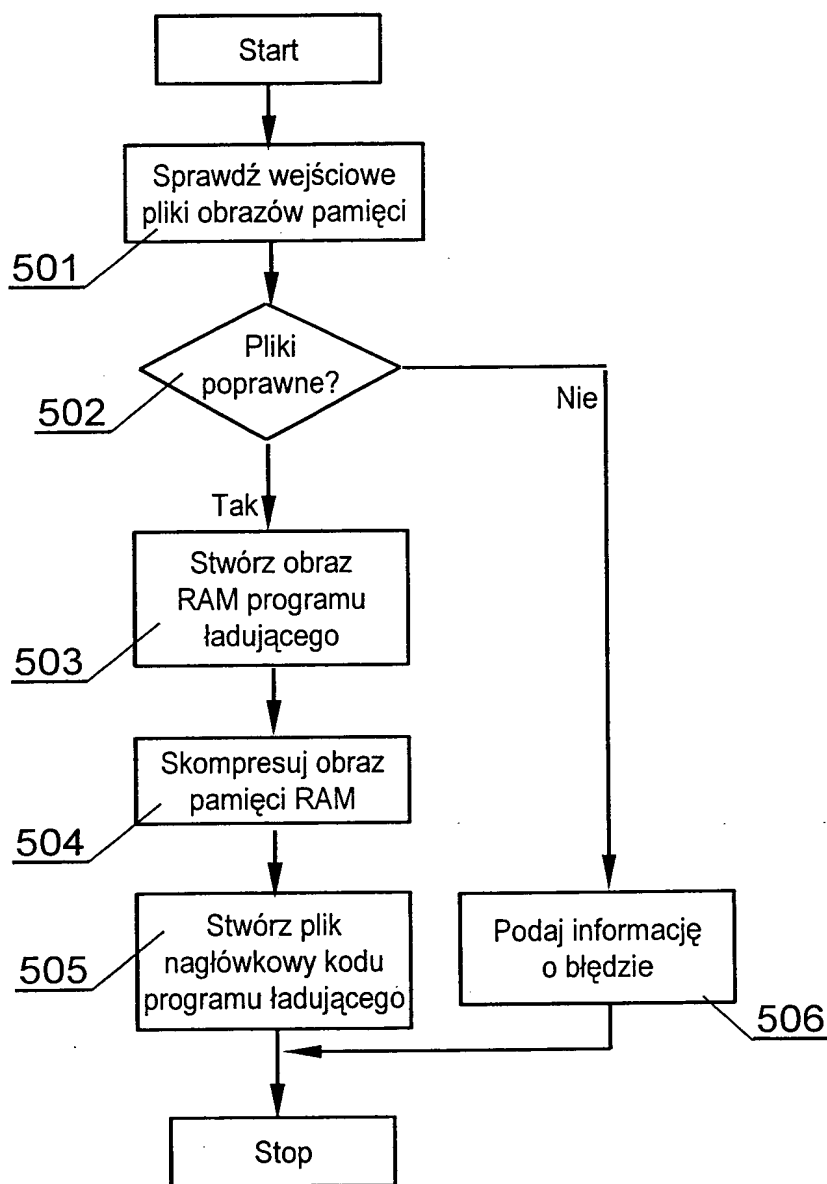


Fig. 5

6 / 7

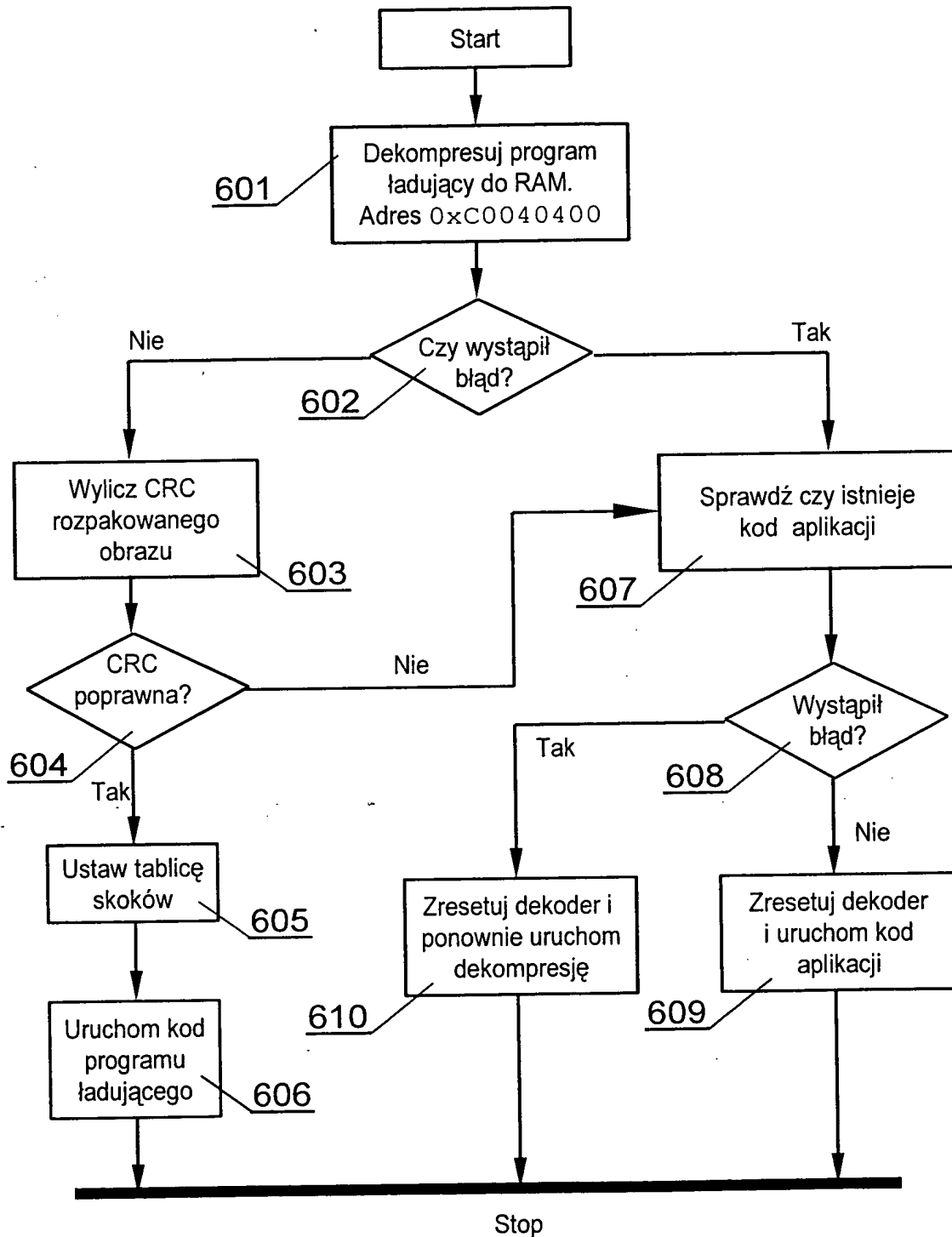


Fig. 6

7 / 7

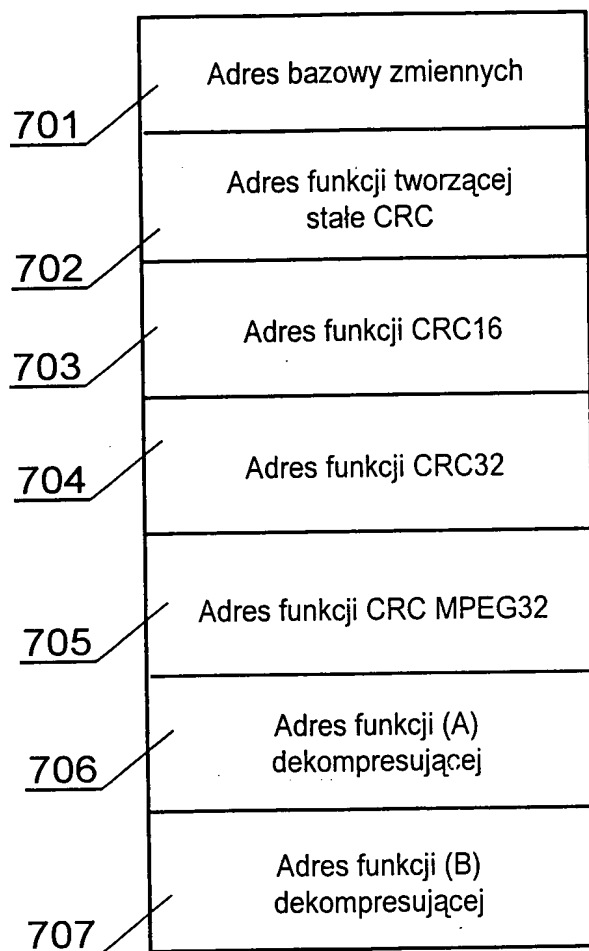



Fig. 7



*Iwona Duma, sworn translator of the English language for the District Court in Warsaw,
2/88 Lachmana Street, 02-786 Warsaw, tel./fax: (22) 855 50 57; mobile: 0 – 608 799 839*

Translation from the Polish language

THE PATENT OFFICE OF THE REPUBLIC OF POLAND

*/in the left margin of the document numbers identifying subsequent paragraphs of the
document/*

/in the middle of the page the national emblem of the Republic of Poland/

A CERTIFICATE

Advanced Digital Broadcast Ltd.,
Taipei, Taiwan

Advanced Digital Broadcast Polska Sp.z o.o.,
Zielona Góra, Poland

on February 14th 2003 submitted to the Patent Office of the Republic of Poland an
application for granting a patent for an invention called „Data signal receiver programmed by the
loader and a method of updating software in a data signal receiver by means of the loader.”





The description of the invention, the patent claims and the drawings, which were attached to this certificate, are true copies of the documents, which were submitted together with the application on February 14th 2003.

The application was submitted under the following number: P-358733.

Warsaw, March 20, 2003

on behalf of the President
Monika Więckowska, MA
Department Director

/-/ illegible signature

/in the left hand bottom corner of the page the impressed round white paper sticker with the national emblem of the Republic of Poland in the middle/

/page 2/

/ in the right hand upper corner of the page the following number:/

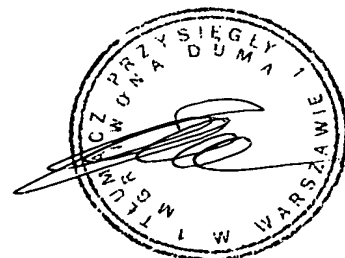
358733

/the number in handwriting:/

3

Data signal receiver programmed by the loader and a method of updating software in a data signal receiver by means of the loader

The object of the invention is a data signal receiver programmed by the loader and a method of updating software in a data signal receiver by means of the loader,



which can be used for example in radio, television or mobile telephony signal receivers, and especially in decoders of digital television.

The loading program called the loader services the procedure of updating software. It processes data packets received by the device and creates a program, which is stored in the FLASH memory of the device.

The state of technique includes a project called EUROLOADER, in which there is a specificity of the loader included for decoders of digital television, accepted by companies associated in the organization called *the European Cable Communication Association*, abbreviated as ECCA. This project describes a method of operation of the loader, which defines the procedure of updating software in decoders. Data related to software updates, according to the specification, are frequently transmitted in order to avoid delays in updating. The applications were divided here into the EUROLOADER program and operating program (application). Moreover the EUROLOADER program is divided into two parts: the starter's and loader's code. The specification of the EUROLOADER program requires from the device to be equipped with at least four types of memory, that is to say memory of RAM, FLASH, ROM and NV-RAM type. The RAM type memory stores temporary information, the FLASH type memory contains the application code, while updating data recorded in the FLASH type memory is possible. In turn, data which are not subject to updating are stored in the ROM type memory,

/page/ 2

including the part of the code of the EUROLOADER program, it is the starter code. The storage of NV-RAM type is mainly dedicated to information about the configuration of the device.

The prior art, known from PC computers includes also programs, used for compression of executable files with '.exe' extension. Their task is to compress programs based on the known solutions of self-extracting archives, which, after a decompression of files into a disk, start the appropriate program.

The essence of the invention is that in the data signal receiver, which is programmed by the loader, including a signal receiving block, the processor, interfaces,



RAM type storage, ROM, NV-RAM and non-volatile memory, the processor has internal block initiating the operation of the loader and a block for servicing the loader. The block for servicing the loader, based on the code, initiated by the initiation block, controls operation of the loader. It is stored in a compressed form in the non-volatile memory, and after being decompressed it is stored in the RAM type memory area, which is declared as a ROM type memory.

Favorably, the memory image is created from the part, including the table of jumps of the loader and a part, including the area of static data of the loader and the code of loader. The compressed memory image is recorded in the non-volatile memory.

Favorably, the jumps table of the loader includes addresses of common functions for the decompressing program and the loader, the functions of which are defined in the decompressing program.

Favorably, the decompressed code of the loader is placed under a fixed address in the RAM memory.

Favorably, the non-volatile memory is a FLASH type memory.

The essence of the invention is in the method of updating the software in the data receiver by means of the loader stored in the non-volatile memory, the receiver also including signal reception block, the processor, interfaces, RAM, ROM, NV-RAM type memories and non-volatile memory

/page/ 3

the updating software is stored in a compressed form in the non-volatile memory, and during the activation of the start procedure it is copied under a fixed address in a part of RAM type memory, which before the process of linking was declared as ROM type memory.

Favorably, the start procedure of the loader is executed at plugging the receiver to the power supply.

Favorably, the start procedure of the loader is started on the request of the user.

Favorably the start procedure of the loader is started by an external signal, send to the receiver.



Favorably the memory image is being created from the part, including the start sequence of the loader, a part containing the jumps table of the loader and a part including the static data area of the loader and the code of the loader, and next the memory image is compressed.

Favorably a table is created, containing the addresses of common functions for the decompressing program and the loader. These functions are defined in the decompressing program.

Favorably, the loader starts the procedure of updating the application by checking if the program, which is currently broadcasted in the stream is dedicated to the receiver, from which the loader was started. It is a condition of updating the application.

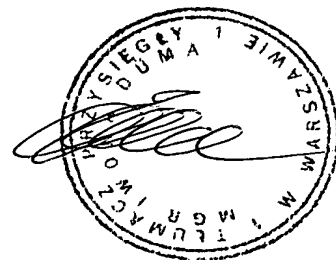
The object of invention is illustrated in the example of embodiment in the drawing, in which fig. 1A shows a digital TV decoder in a simplified form, fig. 1B shows a mobile phone in a simplified version, fig. 2 shows organization of FLASH type memory. Fig. 3A shows the algorithm of starting the uncompressed loader, fig. 3B shows the algorithm of starting the compressed loader, and fig. 4 shows organization of RAM type storage for a compressed loader. Fig. 5 shows the procedure of creating the compressed memory image; fig. 6 shows the procedure of decompressing

/page/ 4

(unpacking) the loader, fig. 7 shows the jumps table of the loader.

The presented method can be applied in various types of signal receivers, of which the digital TV decoder was given only as an example. Such receivers can also include mobile phones, palmtops, radio receivers and all other devices, where software updating is possible through the external signal received by them.

Fig. 1 shows a television signal receiver in a simplified form, which is digital TV receiver 101. The main element of the digital television receiver 101 is a processor 120, which manages the work of the receiver. The processor includes an internal initiating block 121, which initiates the operation of the loader and the block of servicing the loader 122, which on the basis of the code, initiated by the initiating block 121, controls the work of the loader. The processor receives a signal from the MPEG signal reception

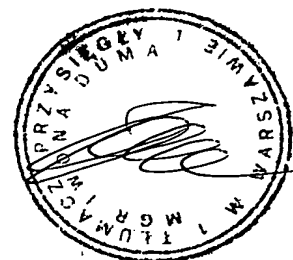


block 110. Additionally the processor 120 has a possibility of bidirectional exchange of data with the remote control unit 131 and transmitting audio/video signal 130 to the receiver, for example a television set. The decoder of digital television includes also a few types of storage, which are biddirectionally linked with the processor. These are FLASH type 150 and RAM type memories 160. These memories store programs controlling the work of the digital TV decoder, including the compressed loader, in which the method of controlling non-volatile memory is recorded, for example FLASH type 150 and RAM 160 of the decoder. The loader is a program, which receives the code of the decoder from external sources and exchanges it in the decoder of digital television, writing it to the FLASH 150 type memory of the digital TV decoder. The loader does not deal with playback of audio or video signals but it is able to communicate with the user of the device displaying information on the screen in order to show the status of the currently performed update of the decoder's software code.

/page/ 5

Another receiver, to which the solution, according to the invention, can be applied, is a mobile phone 211. Such telephone is presented in fig. 1B. Internal elements 221 and 222 of the processor 220, serve the same functions as in case of the television decoder from fig. 1A, alike the memories 250 and 260. However the elements of the receiver change. They serve the function of communication with data sources and communication with the user. The first group of systems includes a signal transmission/reception block GSM 210 and a block of external interfaces 240 for example IrDA. The second group of systems includes the telephone's display control block 230 and keypad control block 231. Similar differences, obvious for a specialist, can be found in other devices, in which the method of this invention would be applied.

The code, which is stored in a decoder of digital television, is divided into two parts: the decoder's code and the loader. The decoder's code, called also a high level code or an application, is responsible for reception, decoding and display of audio, video signals and other components such as teletext or subtitles in different language versions.

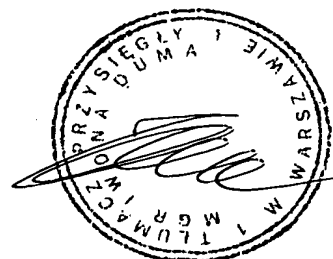


The loader can be started each time, when a digital TV decoder is connected to the power supply. After the start, the loader checks the possible sources of applications. If there is a possibility of updating the application by external decoder's interfaces 140, the loader checks the possibility of updating and if there is such possibility starts to exchange applications. However, if it is possible to exchange applications with the use of signals, broadcasted in a satellite, cable or terrestrial television signal, received by the signal receiving block 110, the loader adjusts itself to the appropriate data stream, using the parameters recorded in the NV-RAM memory. The loader starts the procedure of updating the application, checking if the currently broadcasted program in the stream is dedicated to a digital television decoder, from which the loader was started. If it is so, the updating procedure is accepted.

/page/ 6

The loader can also be started at a request of the user or activated by a specific signal from the side of the broadcaster. The broadcaster can send a signal informing about a new version of software available, and the recipient after receiving this information can issue a request of starting the loader or the loader can be started automatically. Additional information can be sent in this signal, which specifies the source and the type of the sent data. Both, the application code and the code of the loader are stored in the FLASH type memory 150 available in the digital television decoder. Fig. 2 illustrates the organization of FLASH type memory of a digital television decoder, illustrated in fig. 1. FLASH type memory is divided into four parts. The first part 201 includes software of the decoder, and the second part 202 is a free storage area, if such is available. The next part 203 contains optional application data, and the last part is the code of the loader 204. In most cases, the storage area available for the code of the loader is 128 Kilobytes. In the beginning of the start of the loader its whole code is copied to the RAM type memory 160 available in the digital TV decoder.

Fig. 3A illustrates the algorithm of starting the loader. In step 301 the bootstrap is started. It starts to work directly after switching on the power supply of the digital TV decoder or after setting the default values (restart) of the system. The procedure

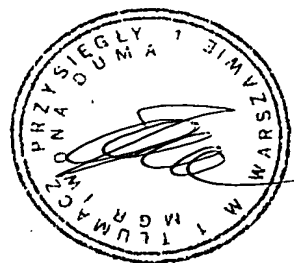


optionally initiates the FLASH type memory depending on the configuration and checks the current status of the digital TV decoder in order to establish, for example, if the loader or application program should be started. In step 302 the initiating part is executed, which depends on the hardware configuration of the decoder, for example the program code is copied from FLASH type memory to RAM type memory. In step 303 the main () function is invoked, which starts the loader, and in step 304 the loader is realized.

Fig. 3B presents the algorithm of starting a compressed loader according to the invention, which is applicable especially in enhanced configurations. In certain configurations the loader's code is so enhanced that it is impossible to place it in the

/page/ 7

128 kilobytes available in the FLASH type memory 150. In order to decrease the capacity of the loader a compression of its code is used. Introduction of the compression of the loader's code influences the increase of the size of available memory for example for enhanced drivers of separate parts of the decoder of digital television such as a modulator or a tuner. In such case the code of the loader consists of: the real application of the loader and the decompressing module (restoring the primary structure of the loader's code) which decompresses the appropriate code of the loader to RAM type storage 160 available in a decoder of digital television. The procedure of starting the program, compressed in this way, is divided into two parts. The first part, which consists of the steps 305, 306, 307, 308, 309, 310, is serviced by the decompressing program. The second part, which consists of step 311, is serviced by the loader. In step 305 the bootstrap is started after the power supply of the digital TV decoder is switched or after initial values of the system are set (restart). In step 306 the code from FLASH type memory to RAM type memory is copied. In step 307 decompression takes place (unpacking, restoring the primary form of the code) of the program to the RAM type memory of the digital TV decoder. In step 308 verification of the decompressed program, it is the code in the primary form, takes place. Next, in step 309 the jumps table, which



was shown in fig. 7, is initiated. In step 310 the main () function, which starts the loader, is executed. In step 311 the loader is started.

Uncompressed loaders can be located in any area of the available memory. In case of a compressed loader constant memory addresses are used, so that the references to the places with variables, constants or pointers to the functions are correct. That is why; it is assumed additionally that the decompressed loader's code is located at a constant address of RAM type memory. In order to prevent the linker from adding procedures, which are normally added when the code is located in a RAM type memory, it is necessary to inform the linker that this area, dedicated to the loader, is a ROM type memory.

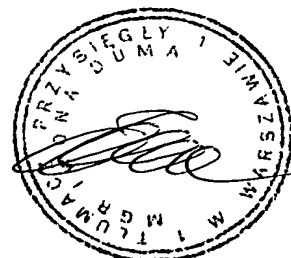
/page/ 8

During the linking of procedures by the linking program all modules of a given program are fetched, information is complemented with references between separate modules, final addresses are dedicated to the functions and the data, for which it was not done at the stage of compilation of the program, and the output program is generated.

Servicing the compressed loader requires a special approach to the operation of placing programs and their data in FLASH type memory. It is assumed that RAM type addresses start with address 0xC0000000 and the size of memory is equal to eight megabytes and four megabytes for a FLASH type memory. Increase of the available memory of RAM type causes that there is an increase of the quantity of available space for the area of general dedication. The addresses of other areas are constant.

Fig. 4 shows organization (division) of the RAM type storage in details. Block 401 represents the area reserved by the loader, that is the memory area reserved for OSGL drivers (abbreviation of *On Screen Graphic for Loader*) and DMUX (abbreviation from *DeMultipleXer submodule*), controlling the filtering of the data, which are dedicated for the loader and come to the decoder of digital television.

Address:	0xC0000000
Size:	256 kB



Block 402 represents a start sequence of the loader that is the memory area, which includes the block of the start code. The address of the first byte from area 402 (0xC0040000) is a start point for the loader.

Address:	0xC0040000
Size:	64 kB

/page/ 9

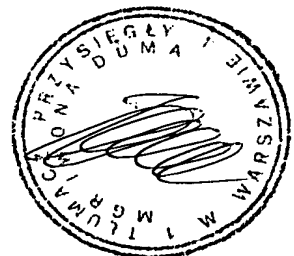
Block 403 represents the memory area reserved by the jumps table of the loader to the common functions for the loader and decompressing module (restoring the primary form of the code).

Address:	0xC0040040
Size:	960 kB

Block 404 represents the memory area occupied by the code of the loader after decompression and by the static data of the loader, for example constants or static variables.

Address:	0xC0040400
Size:	255 kB

Block 405 represents the memory area occupied by the code of the decompressing program and by the static data of the decompressing program, for example constants or static variables.



Address:	0xC0080000
Size:	20 kB

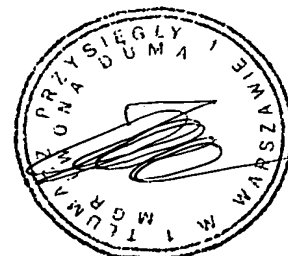
Block 406 represents the area of general dedication, reserved in the memory. The area 406 represents the range of addresses dedicated both to the data of the decompressing program, including the compressed code of the loader and the code of the loader's module.

The use of the compressed loader in the decoders of digital television requires a procedure of creating the RAM memory image.

/page/ 10

In order to perform this procedure after compilation and linking two files are created, including the memory image. In certain systems they are marked with 'hex' extension. The first includes a start sequence of the loader located in block 402; the second is a memory segment including static data and the loader's code, situated in block 404. Blocks 402 and 404 are used to create the image of RAM type memory.

Fig. 5 illustrates the procedure of creating the compressed memory image. After the start, in step 501, files with memory image are read. In step 502 the input files are checked in relation to the memory image. That is if static data and the loader's code are located in the appropriate range of addresses. If the files are incorrect, information about an error is displayed in step 506 and the procedure ends in step 506. In case the files are correct, an image of RAM type memory is created in step 503. Such image constitutes a bytes table comprised sequentially of the areas 402, 403, 404 of the memory. At the stage of creating memory image of RAM type, area 403 is filled with nulls. In step 504 the memory image is compressed, and in step 505 a header file of the loader is created, including a compressed RAM type memory image and information required for correct decompression and planting the code of the loader in the RAM type memory.



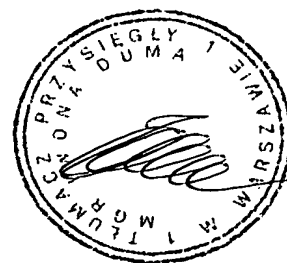
The header file of the loader includes the following options for the decompressing module:

- loader_area_start_addr – address of the location in memory, where the start sequence of the loader starts – block 401;
- loader_area_size – a variable defining the size of the block accessible for the loader;
- loader_code_size - a variable defining the size of the code after the compression operation is executed;
- compressed_loader_crc – a checksum of the compressed code of the loader;
- loader_compression – the selected compression type;
- original_code_addr – the address of the loader's code after decompression – block 404;

/page/ 11

- original_code_size – the size of the code of the loader before compression;
- original_code_crc – a checksum of the loader's code before compression.

Fig. 6 illustrates the procedure of the operation of the program decompressing the loader. The header file of the loader is used by the module of decompressing program a part of which it constitutes. In step 601 decompression of the loader's code to RAM type memory takes place under the address 0x0040400. In step 602 it is checked if an error occurred during the decompression of the code. In case an error occurs, there is a transition to step 607. If there is no error, then in step 603 the CRC checksum of the decompressed loader is calculated and there is a transition to step 604, where a check is performed if the CRC checksum is correct. If the checksum is incorrect a transition to step 607 is made. In case the checksum is correct, first a transition to step 605 is made, where the decompressing program initiates the jumps table of the loader, which is illustrated in fig. 7, and next a transition to step 606 is made. Initiation of the jumps table is a part of the decompressing process and it is performed by filling it up with appropriate addresses. In step 607 the procedure checks, if the application code



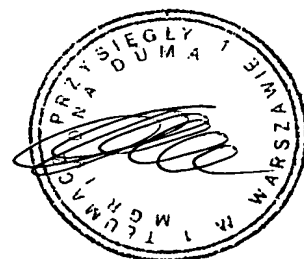
is correct and next in step 608 it checks if there was any error. If no error occurred, initial parameters are set for the digital television decoder in step 609 and the procedure ends its operation starting the application. If there was any error, a transition is made to step 610, where the procedure sets the initial parameters for the decoder. The checksums are counted for memory images before and after the compression, so that the decompressing program can check the correctness of the loader's code before compression and after decompression.

Fig. 7 shows the jumps table of the loader. The table is a set of addresses. Every one of these addresses occupies the space of four bytes. The table is started by a base address 701 of the global static variables. The next addresses are; address 702 of the function initiating the table used for calculating the CRC checksums; address 703 of the function calculating the CRC checksum 16;

/page/ 12

address 704 of the function calculating the CRC checksum 32; address 705 of the function calculating the CRC MPEG32 checksum; address 706 of the A function for decompression of the loader; and address 707 of the B function for decompression of updating data. There are two decompressing functions, because the loader can be compressed with another compression method than the updating data, which are being received.

The application of the compressed loader makes it possible to use a greater quantity of drivers and decreases the required size of the non-volatile memory, for example FLASH type memory, which is more expensive than the RAM type memory when we compare the price per a unit of memory. As opposed to the solutions known from PC computers the application of compression in devices, which do not have a hard disk, is much more difficult because of the requirement of creating appropriate structure of RAM type memory and the use of this structure. Moreover the solution uses memory images and not executable files as it is practiced in the current state of technique.



/oblong stamp with the following contents:/

Power of Attorney

eng. LUDWIK HUDY Maślowski, PhD

Patent Attorney

Reg. No. 3098

/-/ illegible signature

/ in the right hand upper corner of the page the following number:/

358733

/the number in handwriting:/

4

Patent claims

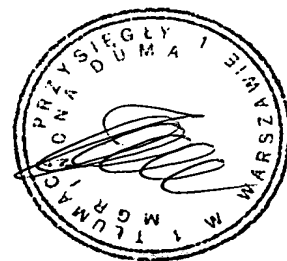
1. The receiver of data signal, programmed by a loader, including a signal receiving block, the processor, interfaces, RAM, ROM, NV-RAM memory type and non-volatile memory, **characterized in that** the processor includes the internal block (121, 122), initiating the work of the loader and the block of servicing the loader (122, 222), which on the basis of the code, initiated by the initiating block (121, 122), controls the operation of the loader, which is recorded in a compressed form in the non-volatile memory (150, 250). After being decompressed, it is recorded in the part of the RAM type memory (160, 260) declared as ROM type memory.
2. The receiver of data signal, according to claim 1, **characterized in that** the memory image is made of the part including the start sequence of the loader, the part including the jumps table of the loader and the part including the area of static data of the loader and the code of the loader and in a compressed form it is recorded in the non-volatile memory (150, 250).



3. The receiver of data signal, according to claim 1, **characterized in that** the jumps table of the loader includes addresses of common functions for the decompressing program and the loader. These functions are defined in the decompressing program.
4. The receiver of data signal, according to claim 1, **characterized in that** the decompressed code of the loader is located under a constant address in the RAM memory.

/page/ 2

5. The receiver of data signal, according to claim 1, **characterized in that** the non-volatile memory (150, 250) is a FLASH type memory.
6. The method of updating software in the data receiver by means of the loader stored in the non-volatile memory, including the signal receiving block, the processor, interfaces, RAM, ROM, NV-RAM memory type and non-volatile memory, characterized in that the updating software is recorded in a compressed form in the non-volatile memory (150, 250) and during the activation of the start procedure it is copied under a constant address in the part of RAM type memory (160, 260), which before the coupling process was declared as a ROM type memory.
7. The method of updating software in a receiver, according to claim 6, characterized in that the start procedure of the loader is executed at the moment of plugging the receiver to the power supply.
8. The method of updating software in a receiver, according to claim 6, characterized in that the start procedure of the loader is activated on the request of the user.



9. The method of updating software in a receiver, according to claim 6, characterized in that the start procedure of the loader is activated by the external signal, being sent to the receiver.
10. The method of updating software in a receiver, according to claim 6, characterized in that the memory image is created from the part including the start sequence of the loader, the part including the jumps table of the loader and the part including the area of static data of the loader and the code of the loader and next the memory image compresses itself.

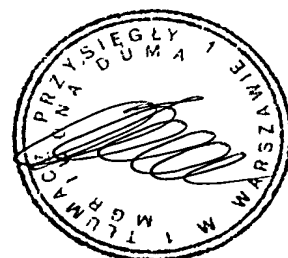
/page/ 3

11. The method of updating software in a receiver, according to claim 6, characterized in that the table is created, including the addresses of the common functions for the decompressing program and the loader. The mentioned functions are defined in the decompressing program.
12. The method of updating software in a receiver, according to claim 6, characterized in that the loader starts the procedure of updating the application by checking if the program, currently broadcasted in the stream is dedicated to the receiver, from which the loader was started, which is a condition for updating the application.

/oblong stamp with the following contents:/

Power of Attorney
eng. LUDWIK HUDY Masłowski, PhD
Patent Attorney
Reg. No. 3098

/-/ illegible signature



/in the subsequent pages in the right hand corner the following number is repeated: /

358733

/the subsequent pages are numbered in handwriting from 5 to 11/

/the subsequent pages are stamped at the bottom of the page with the oblong stamps with the following contents: /

/oblong stamp with the following contents: /

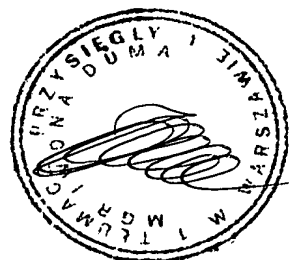
Power of Attorney

eng. LUDWIK HUDY Maślowski, PhD

Patent Attorney

Reg. No. 3098

/-/ illegible signature



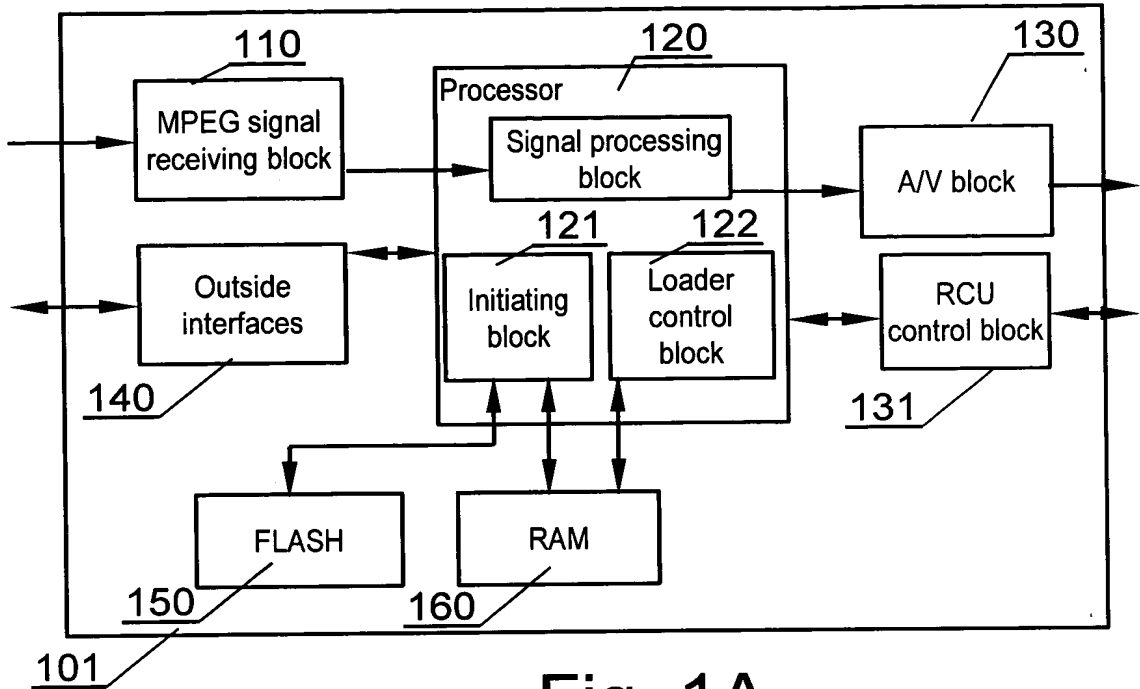


Fig. 1A

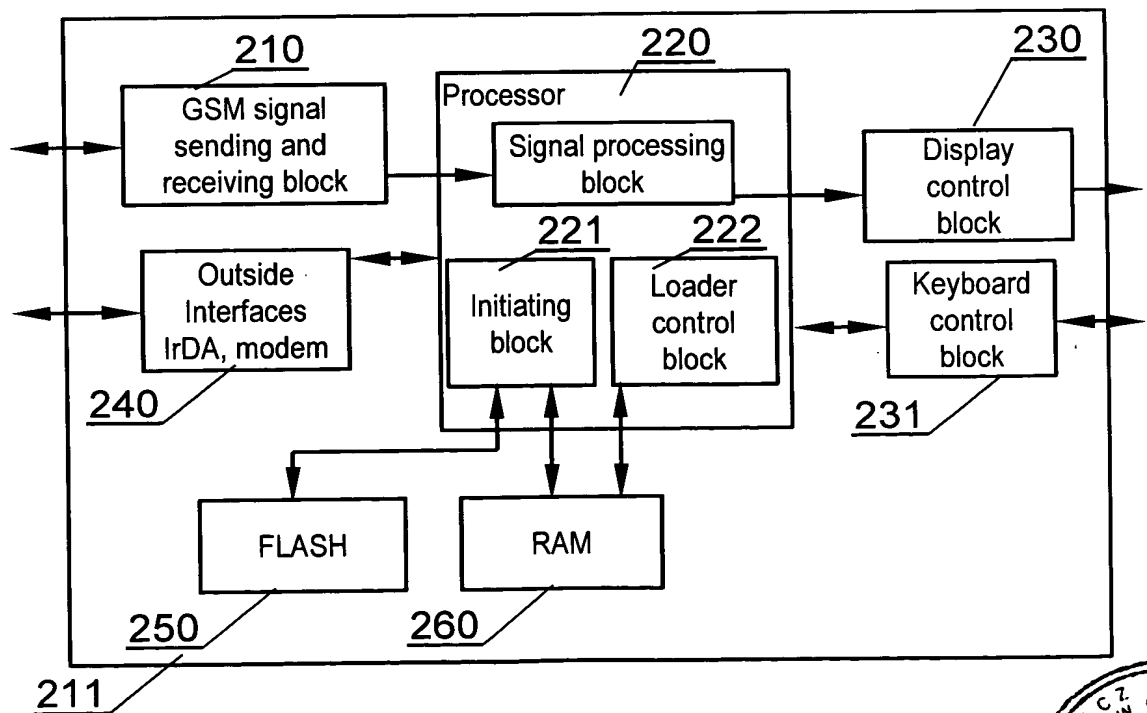


Fig. 1B



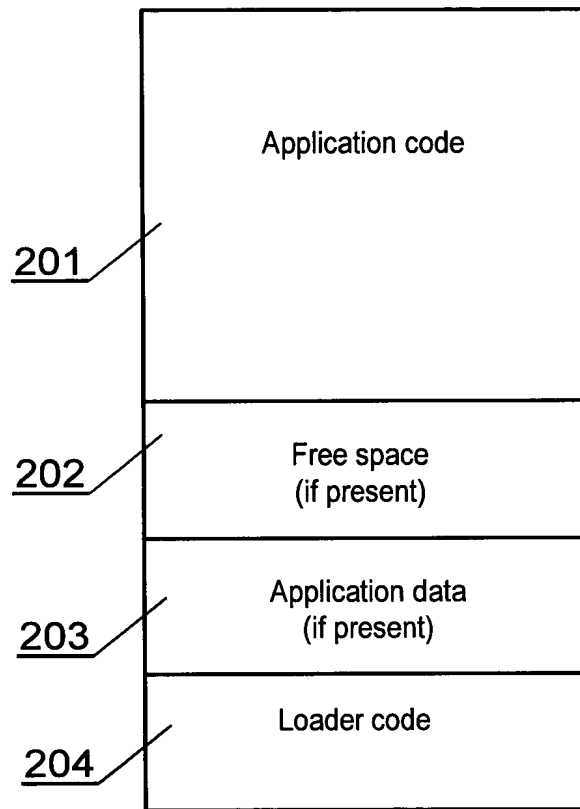


Fig. 2



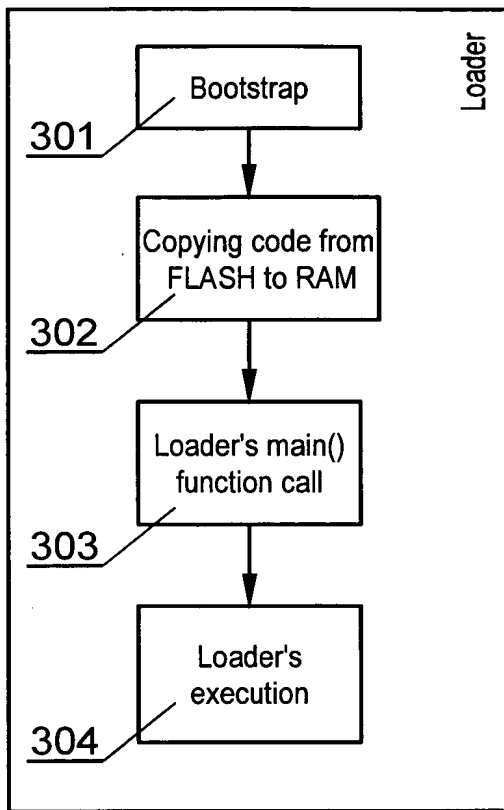


Fig. 3A

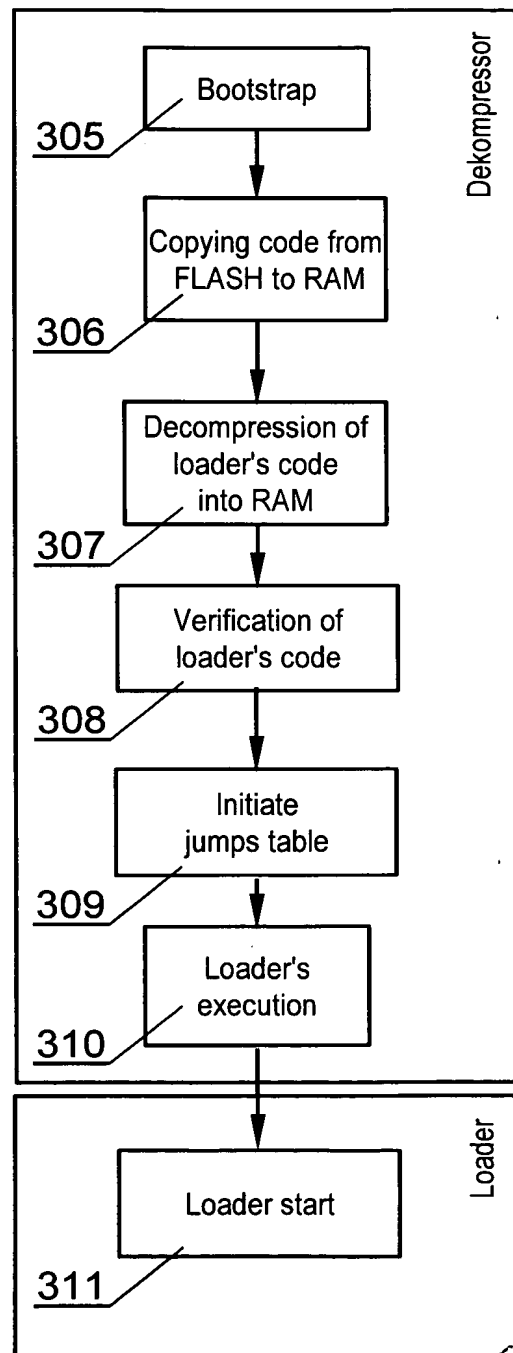
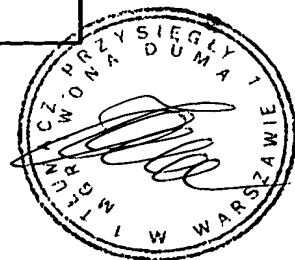


Fig. 3B



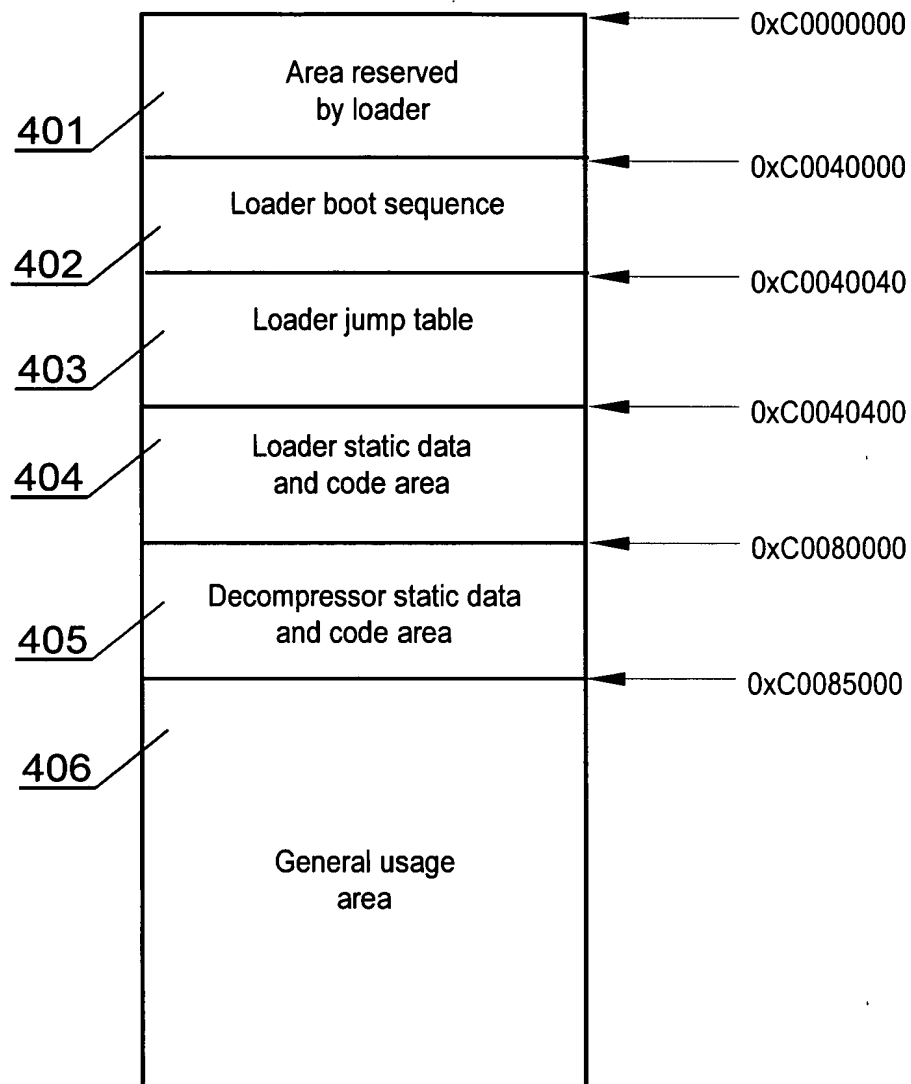


Fig. 4



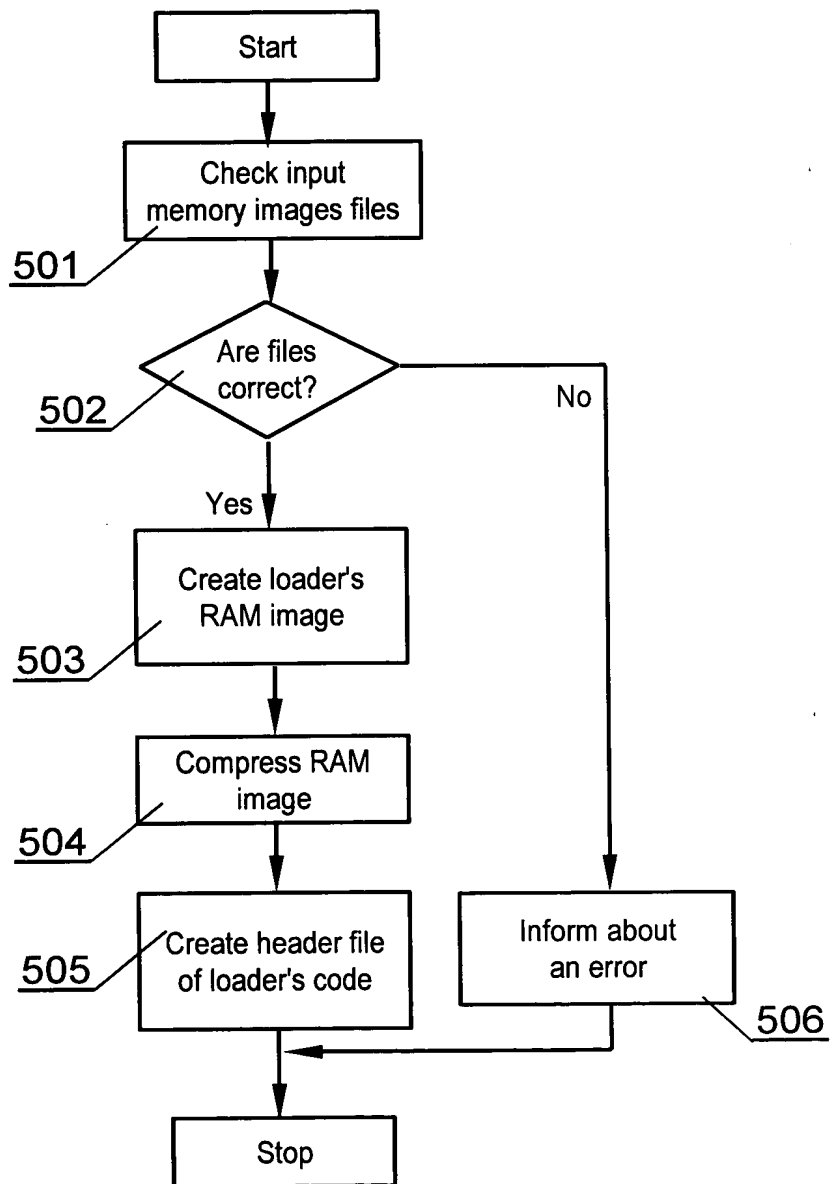


Fig. 5



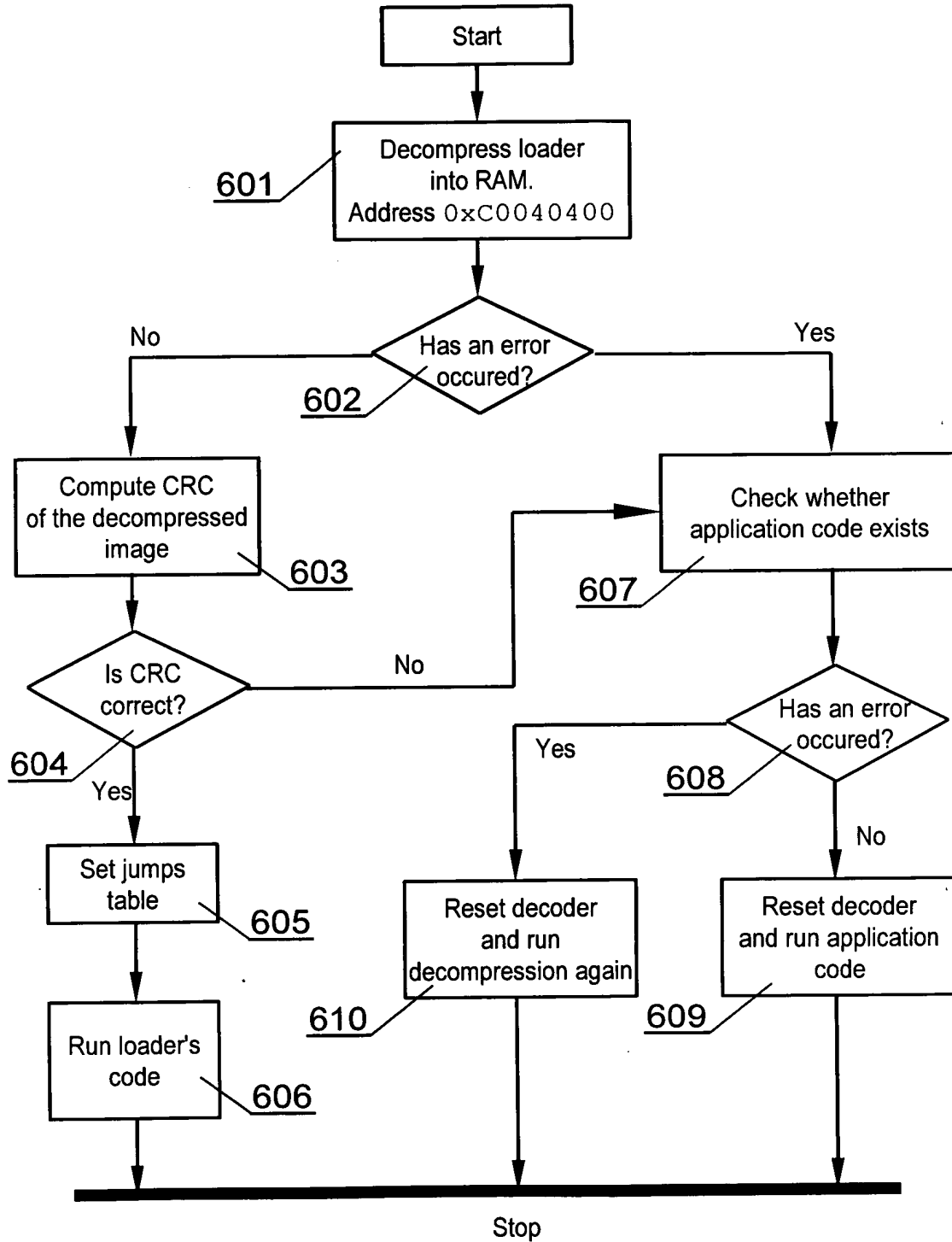
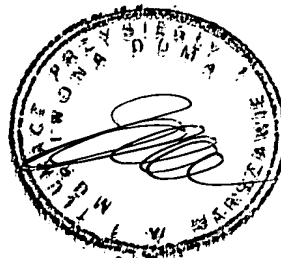


Fig. 6



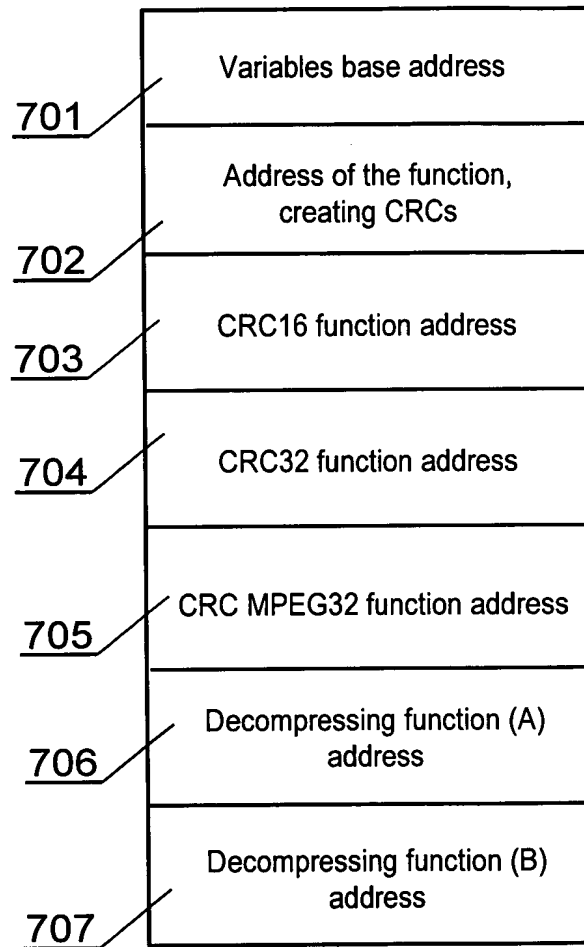
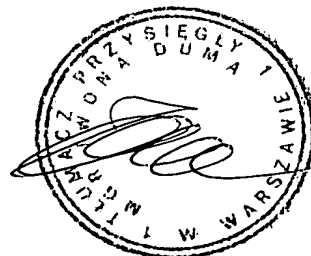


Fig. 7



Repertory No.: 767/12/2003

I, the undersigned, Iwona Duma, sworn translator of the English language for the District Court of the City of Warsaw, hereby certify that the above text is a true and complete translation of the Polish document.

Warsaw, December 18, 2003.

